Final Report of N00014-91-J-1613 Principles of Fault-Tolerant and Efficient Parallel Computation

Paris C. Kanellakis*

June 15, 1994

1 Executive Summary

Motivation: The high-performance potential of parallel and distributed computation can only be realized with significant computation speed-ups from the coordinated action of many processors. A basic problem that has to be addressed, in order to realize this potential, is the unreliability of the resulting (highly complex) systems of many processors. The research of N00014-91-J-1613 "Principles of Fault-Tolerant and Efficient Parallel Computation" has focused, primarily, on the algorithmic principles of fault-tolerant and efficient parallel computing. The desirable combination of reliability and performance is nontrivial since efficiency implies removing redundancy, whereas fault-tolerance requires adding some redundancy to computations.

Accomplishments: We have completed work on both general algorithmic simulation techniques [4, 2, 3, 9, 10, 11, 12] and on interactive software that illustrates these techniques via animation [1]. An overview appears in [5]. We have completed work on self-stabilizing distributed algorithms [6] and studied the complexity of concurrency [7, 8]. Both [4, 6] are significant algorithmic advances, have generated much interest and motivated other work in the larger CS community. These accomplishments realize most of the work proposed in the ONR grant "Principles of Fault-Tolerant and Efficient Parallel Computation". (There are two exceptions. The progress in self-stabilizing protocols [6] was not anticipated in the original proposal, and no progress was made on reliable transaction control problems).

Outline: In Section 2 we summarize the research accomplishments supported by N00014-91-J-1613 entitled "Principles of Fault-Tolerant and Efficient Parallel Computation" (March 31 1991 – September 30 1993). They are grouped in four categories entitled: Robust Parallel Computing, Interactive Animation Software for Robust Algorithms, Robust Distributed Computing, On Concurrent Languages. These categories correspond to analogous sections in the original proposal. We conclude in Section 3 with a review of how the funds were used.

Attached Reports: Copies of the papers mentioned have been already been provided, with the exception of [3, 5, 8]. We include copies for these three publications with this report.

*Department of Computer Science, Brown University, Box 1910, Providence, RI 02912-1910, USA. E-mail: pck@cs.brown.edu; Tel: 401-863-7647; Fax: 401-863-7657.

2 Overview of Research Accomplishments

2.1 Robust Parallel Computing

Our research is an investigation of fault models and parallel computation models under which it is possible to achieve algorithmic efficiency (i.e., speed-ups close to linear in the number of processors) despite the presence of faults. To see that there is a reliability/efficiency trade-off recall that reliability requires *adding redundancy* to the computation in order to detect errors and reassign resources, whereas gaining efficiency by massively parallel computing (i.e., attaining speed-ups close to linear in the number of processors) requires *removing redundancy* from the computation to fully utilize each processor. Specialized trade-offs have been addressed in the area of Algorithm-Based Fault Tolerance (ABFT). Our work can be viewed as a theoretical study of general ABFT on shared memory fail-stop multi-processors. In terms of the theory of parallel algorithms, our work can be viewed as investigating fault-tolerant versions of Brent's fundamental lemma.

It was somewhat surprising when in [4] we demonstrated that it is possible to combine efficiency and fault-tolerance for many basic algorithms expressed as Concurrent Read Concurrent Write Parallel Random Access Machines (the CRCW PRAMs). This work initiated much more research in this area by Kedem, Palem, Spirakis, Rabin, Anderson, Woll and others (published in IEEE FOCS, ACM STOC and ACM PODC).

The [4] fault model is non-trivial since it allows *any pattern of dynamic fail-stop no restart processor errors, as long as one processor remains alive*. This fault model was extended in [10] to include *arbitrary static memory faults, i.e., arbitrary memory initialization*. The CRCW PRAM computation model is a widely used abstraction of a real massively parallel machine. Note that, as shown in [4] it suffices to consider the model where all concurrent writes are identical, called COMMON CRCW PRAM, and that the atomically written words need only contain a constant number of bits. Interestingly, our technique can be extended (as shown in [9]) to *all* CRCW PRAMs (this result was independently derived also by Kedem, Palem and Spirakis). More specifically, in [9] it is shown how to compile any fault-free PRAM into a CRCW PRAM that executes in a fail-stop no restart environment with comparable efficiency. A key algorithmic primitive in all this work is the *Write-All* operation of [4]. Iterated *Write-All* is the basis for the compilation technique in [9] and for the clearing of memory via bootstrapping in [10].

In all the above-mentioned algorithmic work there are two key assumptions that are somewhat unrealistic if compared to more pragmatic reliability analyses. Namely, we assume that: (1) the processors can read and write memory concurrently, and (2) the processor faults are fail-stop without restarts. In [3] we investigate assumption (1) about the model of computation and show how to control and how to minimize this redundancy. Our various analyses are deterministic with worst-case failure-inducing adversaries. For this we still assume (2) above, namely that the dynamic processor errors do not affect shared memory and processors once stopped do not resume the computation. More general processor asynchrony has been examined in by many researchers. Many of these analyses involve average processor behavior and use randomization. Our contributions in removing assumption (2) are included in [2] and are based on deterministic analysis.

Finally, in [5] we present a synthesis of these results with a hierarchy of models and appropriate complexity classes. In [5], we also introduce and solve approximate versions of the *Write-All* problem.

Shared-Memory Architecture: The abstract model that we are studying can be realized in the PRAM architecture in Figure 1. This architecture is more abstract than a realization in terms of hypercubes, but it is simpler to program in. Moreover, fault-tolerant technologies all contribute

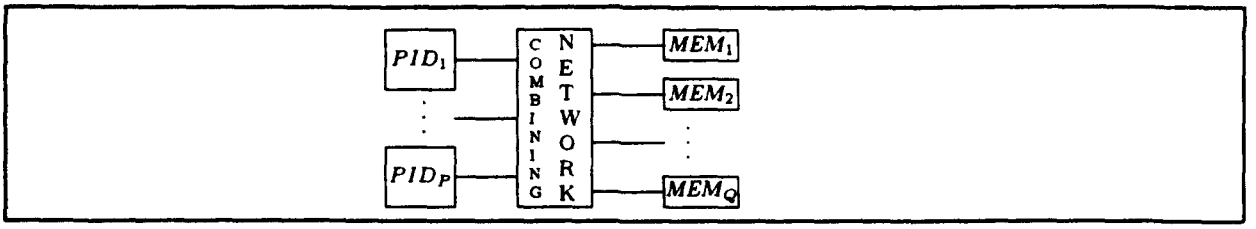

Figure 1: An architecture for a fail-stop multiprocessor.

towards concrete realizations of its components. There are P fail-stop processors. There are Q shared memory cells, the input of size $N \leq Q$ is stored in shared memory. These semiconductor memories can be manufactured with built-in fault tolerance using replication and coding techniques without appreciably degrading performance. Processors and memory are interconnected via a synchronous network (e.g., as in the NYU Ultracomputer) A combining interconnection network that is well suited for implementing synchronous concurrent reads and writes has been developed by Kruskal etal. With this architecture, our algorithmic techniques become applicable; i.e., the algorithms and simulations we develop will work correctly, and within the claimed complexity bounds.

2.2 Interactive Animation Software for Robust Algorithms

Animation of algorithms makes understanding them intuitively easier. As part of this project, Scott Apgar has developed the software tool RAFT (Robust Animator of Fault Tolerant Algorithms) which allows the user to animate a number of efficient fault-tolerant parallel algorithms. The novelty of the system is that the interface allows the user to create new on-line fault-injecting adversaries as the algorithm executes. The various algorithms animated adapt to this interactive input in order to ensure correctness. The system has an extensive user-interface which allows a choice of the number of processors, the number of parallel tasks, the algorithmic techniques, and (most importantly) the interactive adversary controls. It consists of about 6000 lines of C and is built on top of the TANGO animation system developed by John Stasko of Georgia Tech. (8500 lines of C). For a detailed description see [1].

2.3 Robust Distributed Computing

Research has been completed in the area of fault-tolerant distributed computing. The problem addressed (round-robin token management) is a key problem in high-speed networks. The solution proposed in [6] is potentially useful for gigabit networks. It is part of the Ph.D. research of Alain Mayer and was in collaboration with Yoram Ofek and Moti Yung of IBM and Rafail Ostrovsky of MIT. The subject of [6] is a new fault-tolerant round-robin token management scheme, that is self-stabilizing and efficient, i.e., always converges fast to a correct state. More specifically the paper investigates the problem of self-stabilizing, round-robin token management scheme on an anonymous bidirectional ring of identical processors, where each processor is an asynchronous probabilistic (coin-flipping) finite state machine which sends and receives messages. It is shown that the solution to this problem is equivalent to symmetry breaking (i.e., leader election). There have been many subsequent investigations by other researchers on this topic.

Distributed-Memory Architecture: The abstract model that we are studying can be realized in the architecture in Figure 2. This architecture can be implemented in high-speed networks using fast hardware switches (i.e., finite state machines) of size independent of the size of the network.

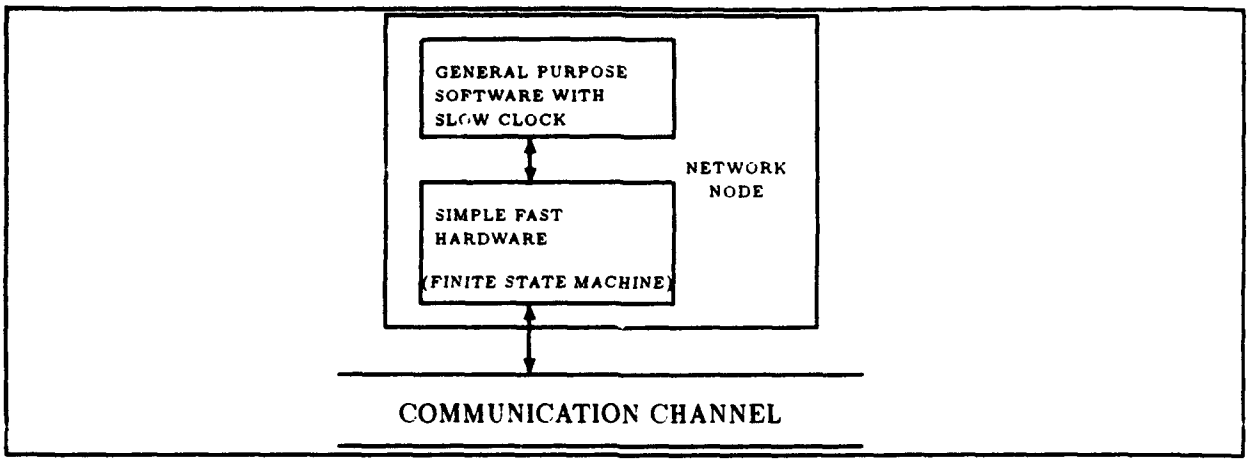

Figure 2: An architecture for a distributed memory model

2.4 On Concurrent Languages

We are in the process of studying how to incorporate our *Write-All* primitives as language constructs in a number of parallel programming formalisms. Our work todate has been at the algorithmic and the abstract machine level. We hope to be able to translate it into primitive language constructs that can be used directly by the programmer.

In the area of theory of concurrent languages, work has been completed on the complexity of interleaving. Interleaving is a very basic operator in the theory of concurrent processes and the completed work is an important addition to the literature on concurrency. It is joint work of Alain Mayer (MSc thesis) and Larry Stockmeyer of IBM [7, 8].

2.5 A Monograph

The last ongoing task in this project is [12]. "Fault-Tolerant and Efficient Parallel Computation" is a forthcoming monograph to be published by Kluwer, (project approved May 1994, to appear in 1995). It will be an augmented version of the PhD thesis of Alex Shvartsman.

3 Conclusions

This final report summarizes the accomplishments under the ONR grant "Principles of Fault-Tolerant and Efficient Parallel Computation" N00014-91-J-1613.

It grant has provided partial support for PhD work (Shvartsman, Michailidis, Mayer) and MSc work (Mayer, Apgar, Michailidis). Also, it has provided two years of summer and travel support for the PI.

References

[1] S.W. Apgar, "Interactive Animation of Fault Tolerant Parallel Algorithms," Brown University Technical Report CS-92-10, 1992. A shortened version appears in the proceedings of the IEEE Workshop on Visual Languages, pp. 11–17, Seattle, Washington, September 1992.

[2] J. Buss, P.C. Kanellakis, P. Ragde, A.A. Shvartsman, "Parallel Algorithms with Processor Failures and Delays," Brown Univ. TR CS-91-54, 1991. Has been accepted subject to minor revision in the *Journal of Algorithms*. (Prel. version appears as P. C. Kanellakis and A. A. Shvartsman, "Efficient Parallel Algorithms On Restartable Fail-Stop Processors," in *Proc. 10th ACM Symp. on Principles of Distributed Comp.*, pp. 23–36, 1991.)

[3] P. C. Kanellakis, Dimitrios Michailidis, and A. A. Shvartsman, "Controlling Memory Access Concurrency in Efficient Fault-Tolerant Parallel Algorithms," Tech report CS-94-23, submitted to the Journal of Parallel and Distributed Computing. Also, the basis for an invited talk at CONCUR94. (Prel. version in *Proc. 7th Inter. Workshop on Distributed Algorithms*, LNCS 725, pp. 99–114, 1993.)

[4] P. C. Kanellakis, and A. A. Shvartsman, "Efficient Parallel Algorithms Can Be Made Robust," in *Distributed Comput.*, vol. 5, pp. 201–217, 1992. (Prel. version in *Proc. 8th ACM PODC*, pp. 138–148, 1989.)

[5] P. C. Kanellakis, and A. A. Shvartsman, "Fault-Tolerance and Efficiency in massively Parallel Algorithms" *Foundations of Ultradependable Parallel and Distributed Computing*, volume II, section 2.2. G. Koob, C. Lau (editors), Kluwer, (to appear in 1994).

[6] A. Mayer, Y. Ofek, R. Ostrovsky, and M. Yung, "Self-Stabilizing Symmetry Breaking in Constant Space," *STOC-92*.

[7] A. Mayer, and L.J. Stockmeyer, "Word Problems – This Time with Interleaving", *IBM RJ 8947 (80223)*, September 1992, (to appear in Information and Computation).

[8] A. Mayer, and L.J. Stockmeyer, "The Complexity of PDL with Interleaving" (to appear in Theoretical Computer Science).

[9] A. A. Shvartsman, "Achieving Optimal CRCW PRAM Fault-Tolerance," in *Info. Processing Letters*, vol. 39, pp. 59–66, 1991.

[10] A. A. Shvartsman, "An Efficient Write-All Algorithm for Fail-Stop PRAM without Initialized Memory," *Information Processing Letters*, pp. 223–231, vol. 44, 1992.

[11] A.A. Shvartsman, "Fault-Tolerant and Efficient Parallel Computation", *Ph.D. thesis*, Brown University Technical Report CS-92-23, 1992

[12] A.A. Shvartsman, P.C. Kanellakis "Fault-Tolerant and Efficient Parallel Computation", forthcoming monograph to be published by Kluwer, (project approved May 1994, to appear in 1995).